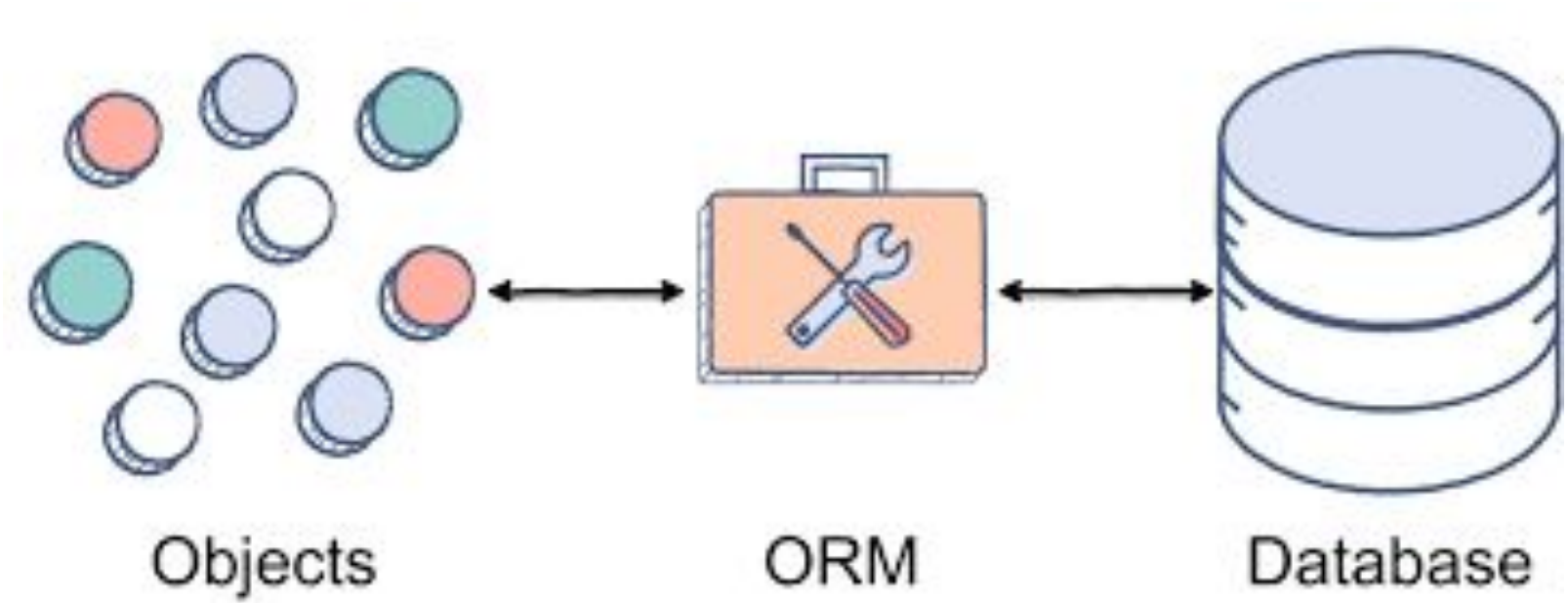
Module 2 - Lecture 7

Database Connectivity

REVIEW

- Database Design
- Normalization
- Data Definition Language
- Data Control Language

-


Objects
ORM
Database

How to connect?

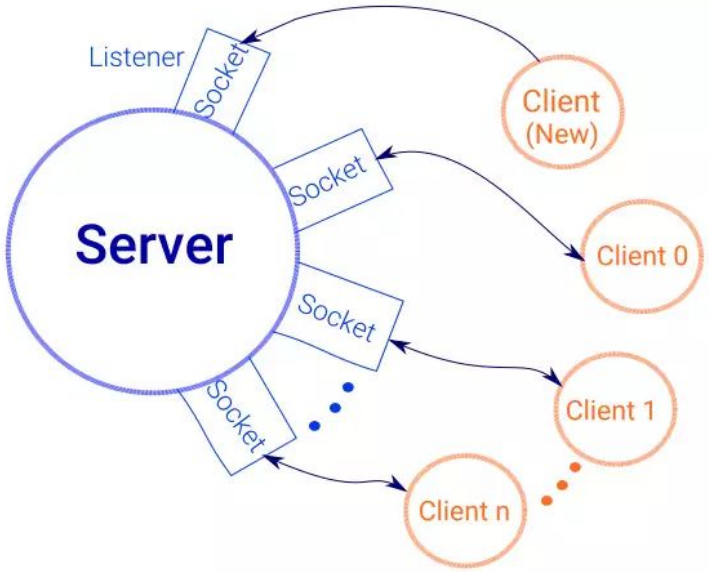
- Our Java applications can act as a client to our database in the same way that pgAdmin and PSQL do.

- We use a common interface provided by the Java language. It is termed Java Database Connectivity, or **JDBC**.

- The things that implement JDBC are called **drivers**. Drivers are written for each database type e.g. PostgreSQL, MSSQL, Oracle, MySQL, etc.

- The PostgreSQL driver we are using is defined in the pom.xml file.

Client / Server programming



Server and Clients

How to connect?

- First, we create a **connection**.
  - Connections use resources.
  - Often, there are a finite number of connections. ~ 100 per GB RAM
  - Connections should be closed, or else.

- We create connections using a **connection string**.
  - This tells the JDBC what it needs to know:
    - Driver to use
    - Database Server URL and port
    - Database to connect to
    - Username / Password to connect with

Anatomy of a Connection String

- Each type of database has its own format.

```
String connectionString =
"jdbc:postgresql://localhost/test?user=oliver&password=secret";
```

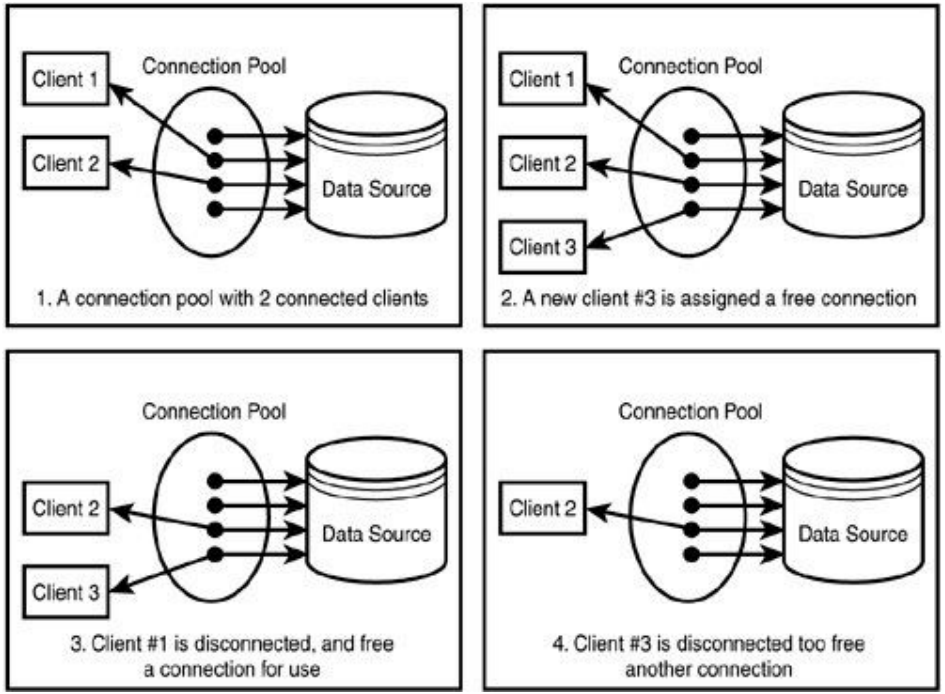
- Connections strings should **NOT** be written in our code!

DataSource

- Our connection strings will be used to create a **DataSource** object.
- **DataSource** is an interface used to define a data source that will be used to establish a connection.
- The implementation of **DataSource** that we will use is **BasicDataSource** supplied by Apache.
- Included as a dependency in pom.xml
- BasicDataSource uses Connection Pooling

Connection Pooling

- There is overhead associated with creating a connection. To limit the cost, we often keep a pool of open connections.



We are connected, now what?

- We can begin to run SQL statements and get back results.

- Our statements should be **parameterized** to prevent SQL Injection attacks!
  - A parameterized query is a query in which placeholders are used for parameters and the parameter values are supplied at execution time.
  - The placeholder in Java is a “?” character.

```
SELECT *
FROM  actor
WHERE first_name = ? AND last_name = ?
```

JDBC Pattern Overview

- There are 4 main interfaces involved.

- **DataSource:** used to configure our connection string and ask for a Connection object.

- **Connection:** used to create SQL queries via Statement objects, and configure transactions.

- **Statement:** used for building and executing static SQL queries.
- **PreparedStatement:** an extension of Statement. Used for building and executing pre-compiled, parameterized queries.

- **ResultSet:** used for parsing the results of a Statement.


there must be an easier way of getting a drink than this!!
ICANHASCHEEZBURGER.COM

JDBCTemplate

- Provided by the popular Spring Java framework.

- Included as a dependency in pom.xml

- We will use two interfaces
  - **JdbcTemplate**: used to connect and execute queries.
  - **SqlRowSet**: used to process rows of results.

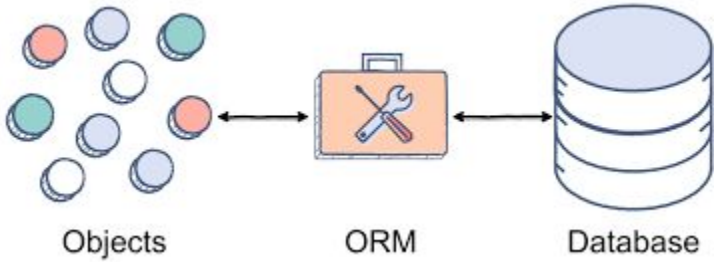
LET'S CODE

Data Access Object (DAO)

- A database table can sometimes map fully or partially to an existing class in Java. This is known as Object-Relational Mapping

- We implement Object Relational Mapping with a design pattern called DAO.

- The pattern uses interfaces so that changes to our data infrastructure results in minimal changes on our business logic.



LET'S CODE

QUESTIONS?